Chapter 1: Problem Set with Solutions

An Overview of Machine Learning and Deep Learning

From "Math and Architectures of Deep Learning"

Problem 1: Input Normalization

A temperature sensor outputs values in the range [-40°C, 120°C].

(a) Using Equation 1.1, normalize a reading of 25°C to the range [0, 1].

(b) If a normalized value is 0.75, what is the original temperature?

(c) Why is normalization important in machine learning?

Solution 1

(a) Using the normalization formula:

$$v_{\text{norm}} = \frac{v - v_{\min}}{v_{\max} - v_{\min}} = \frac{25 - (-40)}{120 - (-40)} = \frac{65}{160} = 0.40625$$

(b) Inverting the normalization formula:

$$v = v_{\text{norm}} \cdot (v_{\max} - v_{\min}) + v_{\min} = 0.75 \times 160 + (-40) = 120 - 40 = 80^{\circ}C$$

(c) Normalization is important because:

- It ensures all features contribute equally during training (prevents features with larger ranges from dominating)
- It improves numerical stability during gradient-based optimization

- It helps the model converge faster during training
- It makes the model less sensitive to the scale of input features

---

Problem 2: Linear Model Computation

Consider the cat brain model with parameters $w_0 = 0.6$, $w_1 = 0.8$, $b = -0.5$.

(a) Compute the threat score for an object with hardness = 0.3 and sharpness = 0.7.

(b) Using threshold $\tau = 0.15$, what decision does the cat make?

(c) Find the equation of the decision boundary (where $y = 0$).

Solution 2

(a) Using Equation 1.3:

$$y = w_0 x_0 + w_1 x_1 + b = 0.6(0.3) + 0.8(0.7) + (-0.5)$$

$$y = 0.18 + 0.56 - 0.5 = 0.24$$

(b) Applying the decision rule (Equation 1.2) with $\tau = 0.15$:

- $y = 0.24 > 0.15 = \tau$

Decision: Run away (threat score exceeds threshold)

(c) The decision boundary occurs where $y = 0$:

$$w_0 x_0 + w_1 x_1 + b = 0$$

$$0.6x_0 + 0.8x_1 - 0.5 = 0$$

$$x_1 = \frac{0.5 - 0.6x_0}{0.8} = 0.625 - 0.75x_0$$

This is a line with slope -0.75 and y-intercept 0.625.

Problem 3: Loss Function Calculation

Given the following training data and predictions:

Instance	Actual (y_gt)	Predicted (y_pred)
1	0.8	0.6
2	-0.3	-0.5
3	0.0	0.2
4	-0.7	-0.6

(a) Calculate the squared error for each instance.

(b) Calculate the total squared error E^2.

(c) Calculate the Mean Squared Error (MSE).

Solution 3

(a) Squared error for each instance: $e_i^2 = (y_{\text{pred}}^{(i)} - y_{\text{gt}}^{(i)})^2$

Instance	Error (y_pred - y_gt)	Squared Error
1	0.6 - 0.8 = -0.2	0.04
2	-0.5 - (-0.3) = -0.2	0.04
3	0.2 - 0.0 = 0.2	0.04
4	-0.6 - (-0.7) = 0.1	0.01

(b) Total squared error:

$$E^2 = \sum_{i=1}^{4} e_i^2 = 0.04 + 0.04 + 0.04 + 0.01 = 0.13$$

(c) Mean Squared Error:

$$\text{MSE} = \frac{1}{N} E^2 = \frac{0.13}{4} = 0.0325$$

Problem 4: Sigmoid Function Properties

(a) Compute σ(0), σ(2), and σ(-2) using Equation 1.5.

(b) Show that σ(-x) = 1 - σ(x). (This is the symmetry property)

(c) What is the derivative of σ(x)? Express it in terms of σ(x) itself.

Solution 4

(a) Using $\sigma(x) = \frac{1}{1+e^{-x}}$:

$$\sigma(0) = \frac{1}{1+e^0} = \frac{1}{1+1} = \frac{1}{2} = 0.5$$

$$\sigma(2) = \frac{1}{1+e^{-2}} = \frac{1}{1+0.1353} \approx \frac{1}{1.1353} \approx 0.881$$

$$\sigma(-2) = \frac{1}{1+e^{2}} = \frac{1}{1+7.389} \approx \frac{1}{8.389} \approx 0.119$$

(b) Proof of symmetry:

$$\sigma(-x) = \frac{1}{1+e^{-(-x)}} = \frac{1}{1+e^{x}}$$

Now consider $1 - \sigma(x)$:

$$1 - \sigma(x) = 1 - \frac{1}{1+e^{-x}} = \frac{1+e^{-x}-1}{1+e^{-x}} = \frac{e^{-x}}{1+e^{-x}}$$

Multiply numerator and denominator by e^x:

$$= \frac{e^{-x} \cdot e^x}{(1+e^{-x}) \cdot e^x} = \frac{1}{e^x+1} = \frac{1}{1+e^x} = \sigma(-x) \quad \checkmark$$

(c) Derivative of σ(x):

Using the chain rule on $\sigma(x) = (1+e^{-x})^{-1}$:

$$\frac{d\sigma}{dx} = -1 \cdot (1+e^{-x})^{-2} \cdot (-e^{-x}) = \frac{e^{-x}}{(1+e^{-x})^2}$$

This can be rewritten as:

$$\frac{d\sigma}{dx} = \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} = \sigma(x) \cdot (1-\sigma(x))$$

Key result: $\sigma'(x) = \sigma(x)(1-\sigma(x))$

This elegant form makes backpropagation computationally efficient!

Problem 5: Geometric Interpretation

Consider a 2D feature space with the separator line $x_0 + x_1 = 1$.

(a) Using the signed distance formula from the textbook, compute the threat score for points:

- P = (0.2, 0.3)
- Q = (0.8, 0.9)
- R = (0.5, 0.5)

(b) Classify each point as "run away," "ignore," or "approach" using τ = 0.2.

(c) On which side of the line does each point lie?

Solution 5

The signed distance from line $x_0 + x_1 - 1 = 0$ is:

$$y = \frac{x_0 + x_1 - 1}{\sqrt{2}}$$

(a) Computing threat scores:

Point P (0.2, 0.3):

$$y_P = \frac{0.2 + 0.3 - 1}{\sqrt{2}} = \frac{-0.5}{1.414} \approx -0.354$$

Point Q (0.8, 0.9):

$$y_Q = \frac{0.8 + 0.9 - 1}{\sqrt{2}} = \frac{0.7}{1.414} \approx 0.495$$

Point R (0.5, 0.5):

$$y_R = \frac{0.5 + 0.5 - 1}{\sqrt{2}} = \frac{0}{1.414} = 0$$

(b) Classification with τ = 0.2:

Point	Threat Score		y	vs τ	Decision
P	-0.354	y < -τ	Approach and purr 😻		
Q	0.495	y > τ	Run away 🏃		
R	0		y	≤ τ	Ignore 😐

(c) Geometric interpretation:

- **P** lies **below** the line (negative distance) → soft/safe region

- **Q** lies **above** the line (positive distance) → hard/dangerous region
- **R** lies **on** the line (zero distance) → boundary

Problem 6: Neural Network Layer Computation

Consider a simple 2-layer network with:

- Input: x = [1, 2]
- Layer 0 weights: $W^{(0)}$ = 0.5, -0.3], [0.2, 0.4 (2×2 matrix)
- Layer 0 has no explicit bias (assume b = 0)
- Activation: sigmoid

(a) Compute the Layer 0 hidden outputs $h^{(0)}$.

(b) If Layer 1 has weights $W^{(1)}$ = [0.6, -0.5] (1×2 matrix), compute the final output.

Solution 6

(a) Layer 0 computation (Equation 1.7):

For each hidden unit j, compute: $z_j = \sum_k w_{jk}^{(0)} x_k$, then $h_j^{(0)} = \sigma(z_j)$

Hidden unit 0:

$$z_0 = w_{00}^{(0)} x_0 + w_{01}^{(0)} x_1 = 0.5(1) + (-0.3)(2) = 0.5 - 0.6 = -0.1$$
$$h_0^{(0)} = \sigma(-0.1) = \frac{1}{1 + e^{0.1}} \approx \frac{1}{1.105} \approx 0.475$$

Hidden unit 1:

$$z_1 = w_{10}^{(0)} x_0 + w_{11}^{(0)} x_1 = 0.2(1) + 0.4(2) = 0.2 + 0.8 = 1.0$$
$$h_1^{(0)} = \sigma(1.0) = \frac{1}{1 + e^{-1}} \approx \frac{1}{1.368} \approx 0.731$$

Layer 0 output: $h^{(0)} \approx [0.475, 0.731]$

(b) Layer 1 computation:

$$z^{(1)} = w_{00}^{(1)} h_0^{(0)} + w_{01}^{(1)} h_1^{(0)} = 0.6(0.475) + (-0.5)(0.731)$$

$$z^{(1)} = 0.285 - 0.366 = -0.081$$

$$h^{(1)} = \sigma(-0.081) \approx \frac{1}{1 + e^{0.081}} \approx \frac{1}{1.084} \approx 0.480$$

Final output: ≈ 0.480

Problem 7: Conceptual Questions

(a) Explain the difference between a regressor and a classifier.

(b) Why do we need nonlinear activation functions in neural networks?

(c) What does "expressive power" mean in the context of neural networks?

(d) Explain why machine learning is described as "function approximation."

Solution 7

(a) Regressor vs. Classifier:

- **Regressor:** Outputs a continuous numerical value (e.g., predicting house price: $425,000)
- **Classifier:** Outputs a discrete class label from a predefined set (e.g., predicting animal type: cat/dog/bird)

In the cat brain example, the threat score estimator is a regressor, while the final run/ignore/approach decision is a classification.

(b) Need for Nonlinear Activations: Without nonlinear activations, a multi-layer network collapses to a single linear transformation:

- If $f_1(x) = W_1x$ and $f_2(x) = W_2x$, then $f_2(f_1(x)) = W_2W_1x = Wx$ (still linear!)
- Nonlinear activations like sigmoid allow the network to learn curved decision boundaries
- This enables the network to approximate complex, nonlinear functions that linear models cannot represent

(c) Expressive Power: Expressive power refers to the class of functions a model can represent:

- A linear model can only represent linear functions (hyperplane boundaries)
- A single sigmoid neuron can represent smooth, monotonic curves
- A deep network with many neurons can approximate arbitrarily complex functions (universal approximation theorem)

Greater expressive power = ability to model more complex relationships in data.

(d) Function Approximation View: Machine learning is function approximation because:

- We don't know the true function f* that maps inputs to outputs
- We have samples: $\{(x_1, y_1), (x_2, y_2), \ldots\}$ where $y_i = f^*(x_i)$
- We create a parameterized model $f(x; \theta)$ that approximates f*
- Training finds parameters θ such that $f(x; \theta) \approx f^*(x)$ on training data
- Success depends on the model having enough expressive power to approximate f* and having sufficient training data to identify the correct parameters

---

Challenge Problem: Derive the Optimal Cat Brain Parameters

Starting from the geometric setup where the optimal decision boundary is the line $x_0 + x_1 = 1$:

(a) Show that the optimal weights are $w_0 = w_1 = 1/\sqrt{2}$ and $b = -1/\sqrt{2}$.

(b) Verify that these parameters give the signed distance from the line.

(c) If we scaled the weights to $w_0 = w_1 = 1$ and $b = -1$, would the classifier still work? Why or why not?

Solution (Challenge)

(a) Deriving optimal parameters:

The signed distance from point (a, b) to line $Ax_0 + Bx_1 + C = 0$ is:

$$d = \frac{Aa + Bb + C}{\sqrt{A^2 + B^2}}$$

For the line $x_0 + x_1 - 1 = 0$, we have A = 1, B = 1, C = -1:

$$d = \frac{x_0 + x_1 - 1}{\sqrt{1^2 + 1^2}} = \frac{x_0 + x_1 - 1}{\sqrt{2}}$$

Comparing with $y = w_0x_0 + w_1x_1 + b$:

$$w_0 = \frac{1}{\sqrt{2}}, \quad w_1 = \frac{1}{\sqrt{2}}, \quad b = \frac{-1}{\sqrt{2}}$$

(b) Verification:

For any point (x_0, x_1):

- $y = (1/\sqrt{2})x_0 + (1/\sqrt{2})x_1 - 1/\sqrt{2} = (x_0 + x_1 - 1)/\sqrt{2}$
- This equals the signed distance formula ✓

(c) Scaled weights analysis:

With $w_0 = w_1 = 1$, $b = -1$:

- $y = x_0 + x_1 - 1$

This is just $\sqrt{2}$ times the original threat score. The decision boundary ($y = 0$) is unchanged: $x_0 + x_1 = 1$.

The classifier would still work because:

- Only the sign and magnitude of y relative to the threshold matter
- Scaling all parameters by the same factor preserves the sign of y
- We would just need to scale our threshold τ by $\sqrt{2}$ as well

This illustrates that the model parameters are not unique—there are infinitely many equivalent parameter sets that define the same decision boundary.

End of Problem Set